

## HW6 : lab2 실습

20160394 임효상

### 1) 실습 A

The screenshot shows a debugger window with the following assembly code:

```
11      str r3, [sp, #4]
12      str r2, [sp, #0]
13
14      mov r6, r0
15      mov r7, r1
16      mov r2, #0
17
18  for1tst:
19      cmp r2, r1
20      bge exit1
21      sub r3, r2, #1
22
23  for2tst:
24      cmp r3, #0
25      blt exit2
26      add r12, r0, r3, LSL #2
27      ldr r4, [r12, #0]
28      ldr r5, [r12, #4]
29      cmp r4, r5
30      ble exit2
31
32      stmb sp!, [r0, r1, r2, r3, r12]
33      mov r0, r6
34      mov r1, r3
35      bl swap
36      ldmb sp!, [r0, r1, r2, r3, r12]
37
38      sub r3, r3, #1
39      b for2tst
40
41  exit2:
42      add r2, r2, #1
43      b for1tst
44
```

The right sidebar shows the memory view and registers. The memory view shows addresses from 0xfffff13c to 0xfffff158. The registers view shows the following values:

name	value (hex)	value (decimal)	data
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x0	0	
r3	0x0	0	

첫 outer loop 수행 전 화면이다. 배열이 7,3,1,2 순서로 저장된 상태다.

The screenshot shows a debugger window with the following assembly code:

```
11      str r3, [sp, #4]
12      str r2, [sp, #0]
13
14      mov r6, r0
15      mov r7, r1
16      mov r2, #0
17
18  for1tst:
19      cmp r2, r1
20      bge exit1
21      sub r3, r2, #1
22
23  for2tst:
24      cmp r3, #0
25      blt exit2
26      add r12, r0, r3, LSL #2
27      ldr r4, [r12, #0]
28      ldr r5, [r12, #4]
29      cmp r4, r5
30      ble exit2
31
32      stmb sp!, [r0, r1, r2, r3, r12]
33      mov r0, r6
34      mov r1, r3
35      bl swap
36      ldmb sp!, [r0, r1, r2, r3, r12]
37
38      sub r3, r3, #1
39      b for2tst
40
41  exit2:
42      add r2, r2, #1
43      b for1tst
44
```

The right sidebar shows the memory view and registers. The memory view shows addresses from 0xfffff13c to 0xfffff158. The registers view shows the following values:

name	value (hex)	value (decimal)	data
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x1	1	
r3	0xffffffff	4294967295	
r4	0x10e28	69160	
r5	0x10158	65880	

첫 번째 외부 루프(i=0) 실행 후 화면이다. j에 해당하는 r3에는 -1이, i에 해당되는 r2에는 1이 저장된다.(i+1을 했으므로) 배열은 첫번째 방이 정렬되어 7,3,1,2다.

show filesystem fetch disassembly reload file jump to line /home/bbo/CA\_2021/lab3/bubblesort.s:43 (56 lines total)

```

11      str r0, [sp, #0]
12      str r3, [sp, #4]
13      str r2, [sp, #8]
14      mov r6, r0
15      mov r7, r1
16      mov r2, #0
17
18  for1tst:
19      cmp r2, r1
20      bge exit1
21      sub r3, r2, #1
22
23  for2tst:
24      cmp r3, #0
25      blt exit2
26      add r12, r0, r3, LSL #2
27      ldr r4, [r12, #0]
28      ldr r5, [r12, #4]
29      cmp r4, r5
30      ble exit2
31
32      stmdb sp!, {r0, r1, r2, r3, r12}
33      mov r0, r6
34      mov r1, r3
35      bl swap
36      ldmbia sp!, {r0, r1, r2, r3, r12}
37
38      sub r3, r3, #1
39      b for2tst
40
41  exit2:
42      add r2, r2, #1
43      b for1tst

```

memory

address	hex	char
0xfffff13c	03 00 00 00	....
0xfffff140	07 00 00 00	....
0xfffff144	01 00 00 00	....
0xfffff148	02 00 00 00	....
0xfffff14c	20 9b 08 00	....
0xfffff150	00 00 00 00	....
0xfffff154	50 09 01 00	P...
0xfffff158	00 00 00 00	....

registers

name	value (hex)	value (decimal)	dr
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x2	2	
r3	0xffffffff	4294967295	
r4	0x7	7	
r5	0x3	3	

두 번째 외부 루프(i=1) 실행 후 화면이다. j에 해당하는 r3에는 -1이, i에 해당되는 r2에는 2가 저장된다.(i+1을 했으므로) 배열은 첫번째 방과 두번째 방까지 정렬되어 3,7,1,2가 된 상태다.

show filesystem fetch disassembly reload file jump to line /home/bbo/CA\_2021/lab3/bubblesort.s:43 (56 lines total)

```

11      str r0, [sp, #0]
12      str r3, [sp, #4]
13      str r2, [sp, #8]
14      mov r6, r0
15      mov r7, r1
16      mov r2, #0
17
18  for1tst:
19      cmp r2, r1
20      bge exit1
21      sub r3, r2, #1
22
23  for2tst:
24      cmp r3, #0
25      blt exit2
26      add r12, r0, r3, LSL #2
27      ldr r4, [r12, #0]
28      ldr r5, [r12, #4]
29      cmp r4, r5
30      ble exit2
31
32      stmdb sp!, {r0, r1, r2, r3, r12}
33      mov r0, r6
34      mov r1, r3
35      bl swap
36      ldmbia sp!, {r0, r1, r2, r3, r12}
37
38      sub r3, r3, #1
39      b for2tst
40
41  exit2:
42      add r2, r2, #1
43      b for1tst

```

memory

address	hex	char
0xfffff13c	01 00 00 00	....
0xfffff140	03 00 00 00	....
0xfffff144	07 00 00 00	....
0xfffff148	02 00 00 00	....
0xfffff14c	20 9b 08 00	....
0xfffff150	00 00 00 00	....
0xfffff154	50 09 01 00	P...
0xfffff158	00 00 00 00	....

registers

name	value (hex)	value (decimal)	dr
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x3	3	
r3	0xffffffff	4294967295	
r4	0x3	3	
r5	0x1	1	

세 번째 외부 루프(i=2) 실행 후 화면이다. j에 해당하는 r3에는 -1이, i에 해당되는 r2에는 3이 저장된다.(i+1을 했으므로) 배열은 첫번째 방과 세번째 방까지 정렬되어 1,3,7,2가 된 상태다.

show filesystem | fetch disassembly | reload file | jump to line | /home/bbo/CA\_2021/lab3/bubblesort.s:43 (56 lines total)

```

11      str r0, [sp, #0]
12      str r3, [sp, #4]
13      str r2, [sp, #8]
14      mov r6, r0
15      mov r7, r1
16      mov r2, #0
17
18  for1tst:
19      cmp r2, r1
20      bge exit1
21      sub r3, r2, #1
22
23  for2tst:
24      cmp r3, #0
25      blt exit2
26      add r12, r0, r3, LSL #2
27      ldr r4, [r12, #0]
28      ldr r5, [r12, #4]
29      cmp r4, r5
30      ble exit2
31
32      stmb sp!, {r0, r1, r2, r3, r12}
33      mov r0, r6
34      mov r1, r3
35      bl swap
36      ldmbia sp!, {r0, r1, r2, r3, r12}
37
38      sub r3, r3, #1
39      b for2tst
40
41  exit2:
42      add r2, r2, #1
43      b for1tst

```

memory

address	hex	char
0xfffff13c	0xfffff15b	4
more		
0xfffff13c	01 00 00 00	....
0xfffff140	02 00 00 00	....
0xfffff144	03 00 00 00	....
0xfffff148	07 00 00 00	....
0xfffff14c	20 9b 08 00	....
0xfffff150	00 00 00 00	....
0xfffff154	50 09 01 00	P...
0xfffff158	00 00 00 00	....
more		

> breakpoints

> signals

registers

name	value (hex)	value (decimal)	dr
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x4	4	
r3	0x0	0	
r4	0x1	1	
r5	0x2	2	

네 번째 외부 루프(i=3) 실행 후 화면이다. j에 해당하는 r3에는 0이, i에 해당되는 r2에는 4가 저장된다.(i+1을 했으므로) 배열은 모두 정렬되어 1,2,3,7가 된 상태다.

r2=r1이므로 ble 조건에 의해 루프가 끝난다.

## 2) 실습 B

Assembly code (lines 14-47):

```

14  mov     r6, r0
15  mov     r7, r1
16  mov     r2, #0
17
18  forltst:
19      cmp     r2, r1
20      bge     exit1
21      sub     r3, r2, #1
22
23  for2tst:
24      cmp     r3, #0
25      blt     exit2
26      add     r12, r0, r3, LSL #2
27      ldr     r4, [r12, #0]
28      ldr     r5, [r12, #4]
29      cmp     r4, r5
30      ble     exit2
31
32      stmdb    sp!, [r0, r1, r2, r3, r12]
33      mov     r0, r6
34      mov     r1, r3
35      bl      swap
36      ldmbia sp!, [r0, r1, r2, r3, r12]
37
38      sub     r3, r3, #1
39      b       for2tst
40
41  exit2:
42      add     r2, r2, #1
43      b       forltst
44
45  exit1:
46      ldr     r2, [sp, #0]
47      ldr     r3, [sp, #4]

```

Registers (r0-r5):

name	value (hex)	value (decimal)	dr
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x2	2	
r3	0xffffffff	4294967295	
r4	0x7	7	
r5	0x3	3	

Memory (0xfffff13c - 0xfffff15b):

address	hex	char
0xfffff13c	03 00 00 00	....
0xfffff140	07 00 00 00	....
0xfffff144	01 00 00 00	....
0xfffff148	02 00 00 00	....
0xfffff14c	20 9b 08 00	....
0xfffff150	00 00 00 00	....
0xfffff154	50 09 01 00	P...
0xfffff158	00 00 00 00	....

add r2,r2,#1 명령어에 의해 r2(i)가 2로 바뀐 시점이다.

Assembly code (lines 14-47):

```

14  mov     r6, r0
15  mov     r7, r1
16  mov     r2, #0
17
18  forltst:
19      cmp     r2, r1
20      bge     exit1
21      sub     r3, r2, #1
22
23  for2tst:
24      cmp     r3, #0
25      blt     exit2
26      add     r12, r0, r3, LSL #2
27      ldr     r4, [r12, #0]
28      ldr     r5, [r12, #4]
29      cmp     r4, r5
30      ble     exit2
31
32      stmdb    sp!, [r0, r1, r2, r3, r12]
33      mov     r0, r6
34      mov     r1, r3
35      bl      swap
36      ldmbia sp!, [r0, r1, r2, r3, r12]
37
38      sub     r3, r3, #1
39      b       for2tst
40
41  exit2:
42      add     r2, r2, #1
43      b       forltst
44
45  exit1:
46      ldr     r2, [sp, #0]
47      ldr     r3, [sp, #4]

```

Registers (r0-r5):

name	value (hex)	value (decimal)	dr
r0	0xfffff13c	4294897980	
r1	0x1	1	
r2	0x2	2	
r3	0x1	1	
r4	0x7	7	
r5	0x1	1	

Memory (0xfffff13c - 0xfffff15b):

address	hex	char
0xfffff13c	03 00 00 00	....
0xfffff140	07 00 00 00	....
0xfffff144	01 00 00 00	....
0xfffff148	02 00 00 00	....
0xfffff14c	20 9b 08 00	....
0xfffff150	00 00 00 00	....
0xfffff154	50 09 01 00	P...
0xfffff158	00 00 00 00	....

i=2인 outer loop에서 첫 번째 inner loop로 j=1인 상태이다. r4는 배열의 두번째 방(7), r5는 배열의 세번째 방(1)을 가리키고 있다. cmp이후 ble에서 branch하지 않고 swap함수를 거치게된다.

show filesystem fetch disassembly reload file jump to line /home/bbo/CA\_2021/lab3/bubblesort.s:36 (156 lines total)

```

14      mov     r6, r0
15      mov     r7, r1
16      mov     r2, #0
17
18  for1tst:
19      cmp     r2, r1
20      bge     exit1
21      sub     r3, r2, #1
22
23  for2tst:
24      cmp     r3, #0
25      blt     exit2
26      add     r12, r0, r3, LSL #2
27      ldr     r4, [r12, #0]
28      ldr     r5, [r12, #4]
29      cmp     r4, r5
30      ble     exit2
31
32      stmdb   sp!, {r0, r1, r2, r3, r12}
33      mov     r0, r6
34      mov     r1, r3
35      bl      swap
36      ldmbia sp!, {r0, r1, r2, r3, r12}
37
38      sub     r3, r3, #1
39      b       for2tst
40
41  exit2:
42      add     r2, r2, #1
43      b       for1tst
44
45  exit1:
46      ldr     r2, [sp, #0]
47      ldr     r3, [sp, #4]

```

memory

address	hex	char
0xfffff13c	03 00 00 00	....
0xfffff140	01 00 00 00	....
0xfffff144	07 00 00 00	....
0xfffff148	02 00 00 00	....
0xfffff14c	20 9b 00 00	....
0xfffff150	00 00 00 00	....
0xfffff154	50 09 01 00	P...
0xfffff158	00 00 00 00	....

breakpoints  
signals

registers

name	value (hex)	value (decimal)	di
r0	0xfffff13c	4294897980	
r1	0x1	1	
r2	0x7	7	
r3	0x1	1	
r4	0x7	7	
r5	0x1	1	

swap함수를 거친 상태다. 두번째 방이었던 7과 세번째 방이었던 1이 swap된 상태이다.

swap함수를 보면 레지스터 r2,r3를 변수로 사용하고 있다. 따라서 r2에는 두번째 방의 7, r3에는 세번째 방의 1이 저장된 것을 확인할 수 있다.

r1과 r0 역시 배열의 인덱스를 swap함수에서 사용하기위해 이전에 r0는 배열 시작주소, r1은 swap할 인덱스로 초기화 된 것을 볼 수 있다.

show filesystem fetch disassembly reload file jump to line /home/bbo/CA\_2021/lab3/bubblesort.s:38 (156 lines total)

```

11      str     r0, [sp, #0]
12      str     r3, [sp, #4]
13      str     r2, [sp, #0]
14
15      mov     r6, r0
16      mov     r7, r1
17      mov     r2, #0
18
19  for1tst:
20      cmp     r2, r1
21      bge     exit1
22      sub     r3, r2, #1
23
24  for2tst:
25      cmp     r3, #0
26      blt     exit2
27      add     r12, r0, r3, LSL #2
28      ldr     r4, [r12, #0]
29      ldr     r5, [r12, #4]
30      cmp     r4, r5
31      ble     exit2
32
33      stmdb   sp!, {r0, r1, r2, r3, r12}
34      mov     r0, r6
35      mov     r1, r3
36      bl      swap
37      ldmbia sp!, {r0, r1, r2, r3, r12}
38
39      sub     r3, r3, #1
40      b       for2tst
41
42  exit2:
43      add     r2, r2, #1
44      b       for1tst

```

threads  
local variables  
expressions  
Tree

memory

start address end address ( 4 )

no memory to display

breakpoints

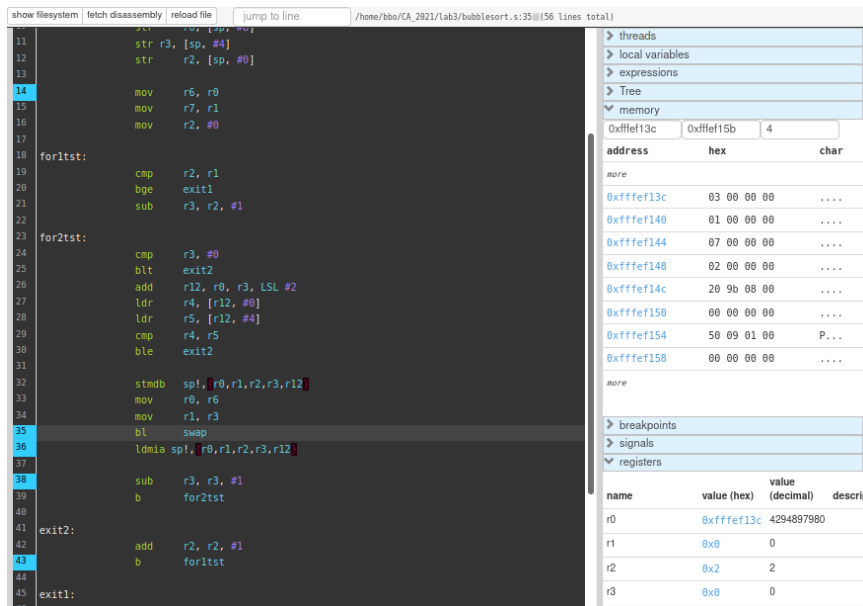
signals

send SIGINT to  
gdb (pid 4368) debug program (pid 1)  
other pid pid

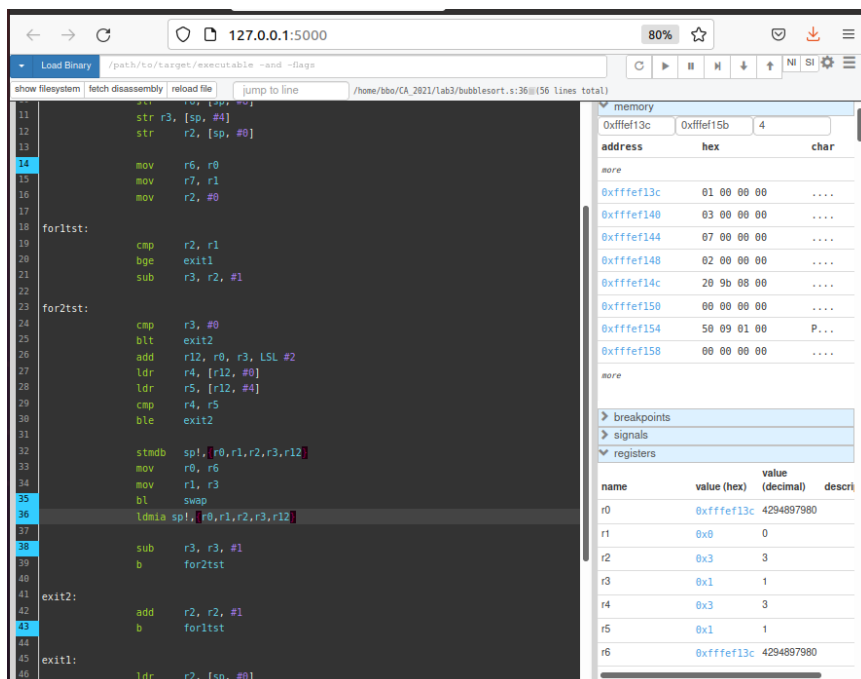
registers

name	value (hex)	value (decimal)	di
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x2	2	
r3	0x1	1	
r4	0x7	7	
r5	0x1	1	
r6	0xfffff13c	4294897980	
r7	0x4	4	
r8	0x0	0	re (6)
r9	0x0	0	re

스택에 저장해두었던 데이터를 이용하여 swap함수로 진입하기 전 레지스터 상태를 복구했다.



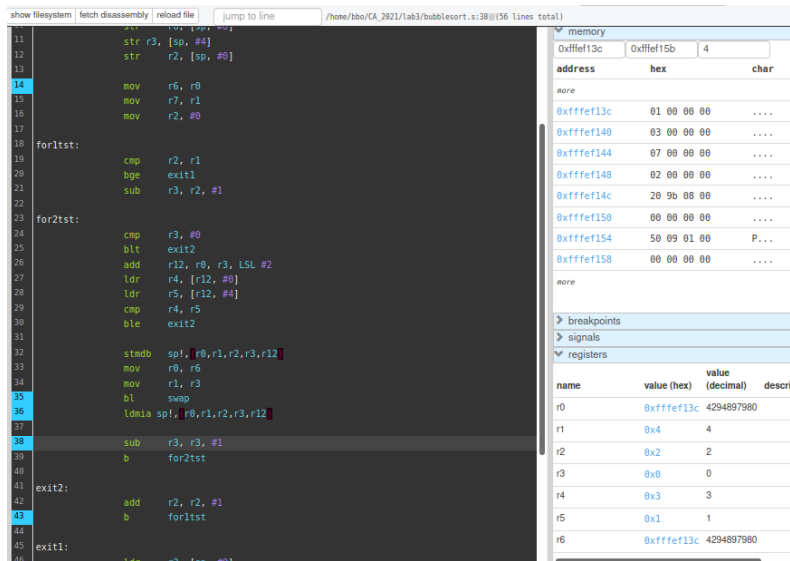
inner loop가 1회 끝나고 r3가 1 감소하여 0이 되었다. 즉 j=0이다. r4는 배열의 첫 번째 방(3), r5는 배열의 두 번째 방(1)을 가리키고 있다. cmp이후 ble에서 branch하지 않고 swap함수를 거치게 된다.



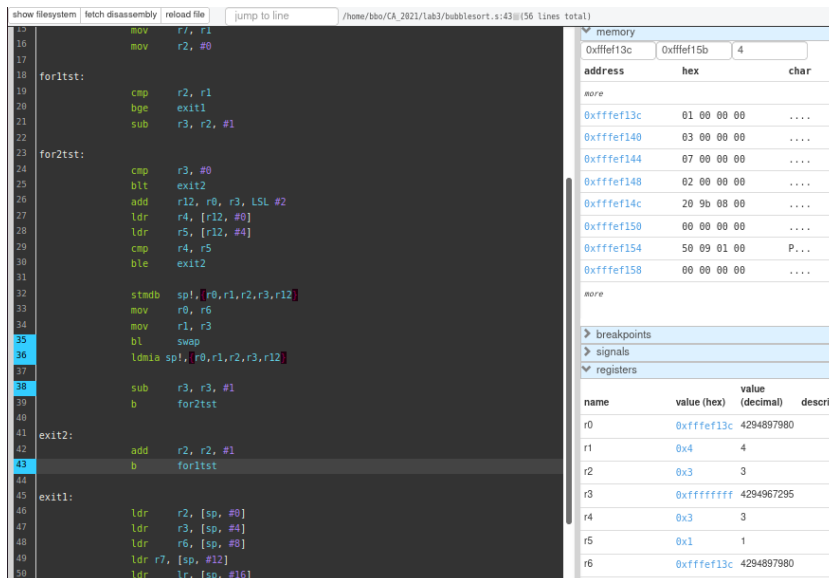
swap함수를 거친 상태다. 첫 번째 방이었던 3과 두 번째 방이었던 1이 swap된 상태이다.

swap함수에서 사용하기 위해 r2에는 첫 번째 방의 3, r3에는 두 번째 방의 1이 저장된 것을 확인할 수 있다.

r1과 r0 역시 배열의 인덱스를 swap함수에서 사용하기 위해 이전에 r0는 배열 시작주소, r1은 swap할 인덱스로 초기화 된 것을 볼 수 있다.

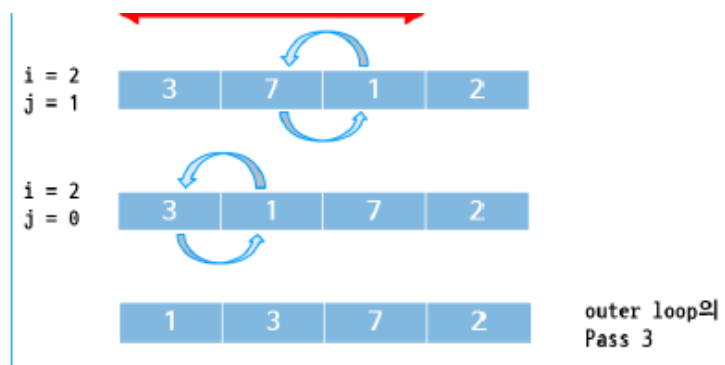


스택에 저장해두었던 데이터를 이용하여 swap함수로 진입하기 전 레지스터 상태를 복구했다.



r3에서 1을 뺀 상태로, r3=-1이므로, cmp r3,#0와 blt 에서 branch된 모습이다.

inner loop가 모두 끝나고 outer loop가 1회 종료되면서 r2는 1이 더해서 3이 된다.



arr에 제대로 저장된 것을 확인할 수 있다.

### 3) 실습 C

```

1  .text
2  .global bubblesort
3  .global swap
4  .type bubblesort, STT_FUNC
5
6  bubblesort:
7      sub    sp, sp, #20
8      str    lr, [sp, #16]
9      str    r7, [sp, #12]
10     str    r6, [sp, #8]
11     str    r3, [sp, #4]
12     str    r2, [sp, #0]
13
14     mov    r6, r0
15     mov    r7, r1
16     mov    r2, #0
17
18 for1st:
19     cmp    r2, r1
20     bge    exit1
21     sub    r3, r2, #1
22
23 for2tst:
24     cmp    r3, #0
25     blt    exit2
26     add    r12, r0, r3, LSL #2
27     ldr    r4, [r12, #0]
28     ldr    r5, [r12, #4]
29     cmp    r4, r5
30     ble    exit2
31
32     ;stmdb sp!, {r0, r1, r2, r3, r12}
33     mov    r0, r6
34     mov    r1, r3

```

name	value (hex)	value (decimal)	dr
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x8b390	570256	
r3	0xfffff13c	4294897980	
r4	0x10e28	69160	
r5	0x10158	65880	
r6	0xfffff13c	4294897980	
r7	0x10158	65880	
r8	0x0	0	re

초기화면이다. arr에 7,3,1,2가 순서대로 들어있다.

```

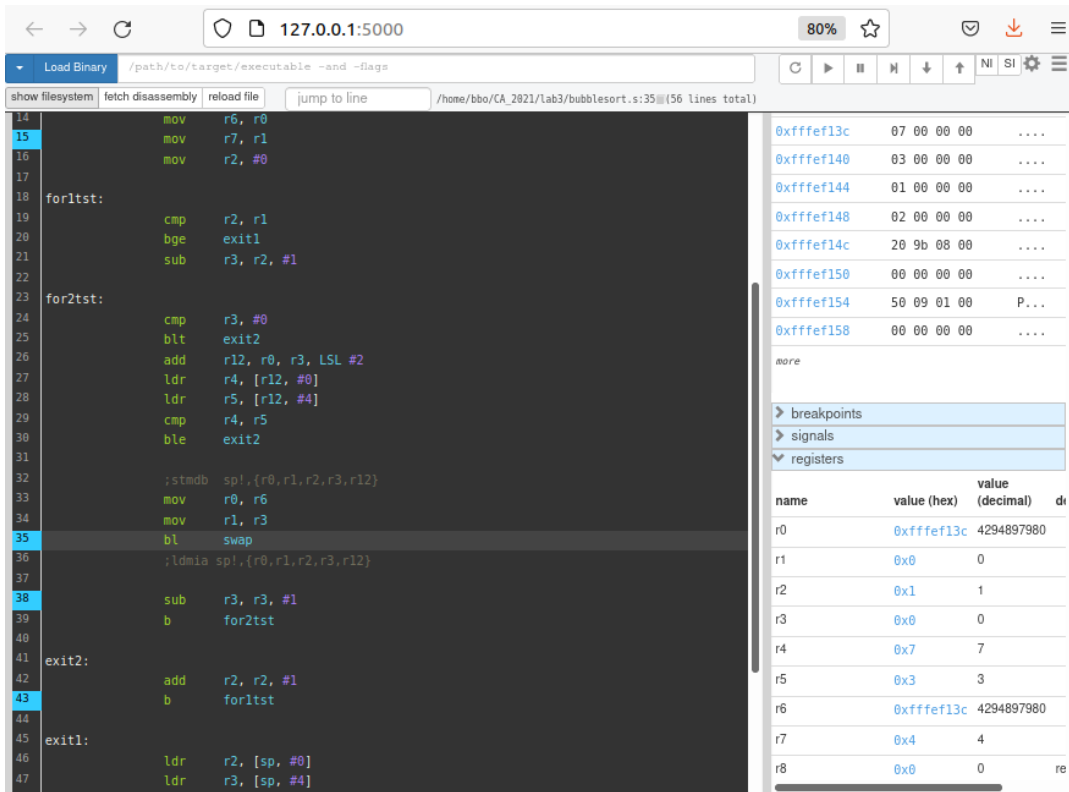
24     cmp    r3, #0
25     blt    exit2
26     add    r12, r0, r3, LSL #2
27     ldr    r4, [r12, #0]
28     ldr    r5, [r12, #4]
29     cmp    r4, r5
30     ble    exit2
31
32     ;stmdb sp!, {r0, r1, r2, r3, r12}
33     mov    r0, r6
34     mov    r1, r3
35     bl     swap
36     ;ldmia sp!, {r0, r1, r2, r3, r12}
37
38     sub    r3, r3, #1
39     b      for2tst
40
41 exit2:
42     add    r2, r2, #1
43     b      for1st
44
45 exit1:
46     ldr    r2, [sp, #0]
47     ldr    r3, [sp, #4]
48     ldr    r6, [sp, #8]
49     ldr    r7, [sp, #12]
50     ldr    lr, [sp, #16]
51     add    sp, sp, #20
52
53     mov    pc, lr
54
55 .end
56

```

name	value (hex)	value (decimal)	dr
r0	0xfffff13c	4294897980	
r1	0x4	4	
r2	0x1	1	
r3	0xffffffff	4294967295	
r4	0x10e28	69160	
r5	0x10158	65880	
r6	0xfffff13c	4294897980	
r7	0x4	4	
r8	0x0	0	re

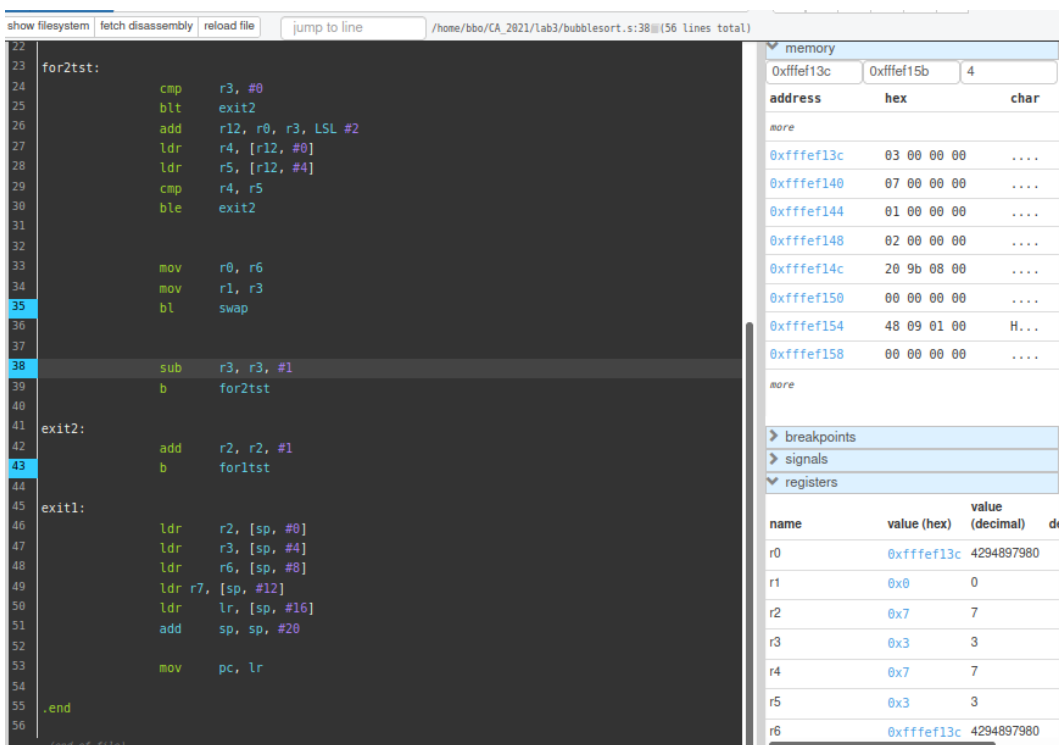
r2의 초기값은 0이었다. 따라서 outer loop에서 r3가 -1로 설정되고 inner loop에서 blt exit2로 branch되었다. add r2,r2,#1 명령어로 현재 r2는 1이 된 상태이다.





swap함수에 들어가기 전 상태이다.

r1과 r0는 각각 배열의 인덱스를 swap함수에서 사용하기 위해 이전에 r0는 배열 시작주소, r1은 swap할 인덱스로 초기화 된 것을 볼 수 있다.



swap함수를 거친 상태다. 배열의 첫번째 방이었던 7과 배열의 두번째 방이었던 3이 서로 swap된

모습을 볼 수 있다.

r2, r3의 경우 각각 배열의 첫번째 데이터 7, 두번째 데이터 3이 된 것을 확인할 수 있다.

r0-r3, r12는 caller save로 호출하는 함수에서 호출 전 레지스터 상태를 저장해야 한다. 이 값들은 callee에서 원래 상태로 복구를 보장해주지 않기 때문이다. 이에 따라 swap함수에서는 r2, r3를 배열에 들어있는 데이터 값으로 활용하였고, bubblesort함수에서는 이전 상태를 저장해두고 복구하지 않아 swap함수에서 사용된 값이 그대로 넘어온 것을 확인할 수 있다.

#### 4) 실습 D

##### 변경된 코드

```
1  .text
2  .global bubblesort
3  .global swap
4  .type bubblesort, STT_FUNC
5
6  bubblesort:
7      sub    sp, sp, #24
8      str    lr, [sp, #20]
9      str    r8, [sp, #16]
10     str    r7, [sp, #12]
11     str    r6, [sp, #8]
12     str    r3, [sp, #4]
13     str    r2, [sp, #0]
14
15     mov     r6, r0
16     mov     r7, r1
17     mov     r2, #0
18     sub     r1, r1, #1
19
20 for1tst:
21     cmp     r2, r1
22     bge     exit1
23     mov     r3, #0
24     sub     r8, r1, r2
25
26 for2tst:
27     cmp     r3, r8
28     bge     exit2
29     add     r12, r0, r3, LSL #2
30     ldr     r4, [r12, #0]
31     ldr     r5, [r12, #4]
32     cmp     r4, r5
33     ble     exit2
34
35     stmdb   sp!, {r0, r1, r2, r3, r12}
36     mov     r0, r6
37     mov     r1, r3
38     bl      swap
39     ldmbia sp!, {r0, r1, r2, r3, r12}
40
41     add     r3, r3, #1
42     b       for2tst
43
44 exit2:
45     add     r2, r2, #1
46     b       for1tst
47
48 exit1:
49     ldr     r2, [sp, #0]
50     ldr     r3, [sp, #4]
51     ldr     r6, [sp, #8]
52     ldr     r7, [sp, #12]
53     str     r8, [sp, #16]
54     ldr     lr, [sp, #20]
55     add     sp, sp, #24
56
57     mov     pc, lr
58
59 .end
60
61 (end of file)
```

결과

```
bbo@ubuntu:~/CA_2021/lab3$ qemu-arm -g 8080 ./lab3
Array before sorting : 7 3 1 2
Array after sorting : 1 2 3 7
bbo@ubuntu:~/CA_2021/lab3$
```